LISTINGS NEWSLETTER

Newsletter of the
Long Island Sinclair/Timex
Users Group

Incorporating NYTSE

Issue JANUARY 1991

IN MEMORIAL.

Last month I had the sad duty to report the death of Warren Brocke.

Warren was respected throughout the TIMEX comunity, and his articles appeared in publications such as Time Designs and Listings.

This issue of Listings is dedicated in honor, respect, and memory of Warren.

Table of Contents

LIST OFFICERS

PRES. HARVEY RAIT
TRES. ROBERT MALLOY
COR.SEC. JOHN PAZMINO
EDITOR. FRED STERN
LIBR. TOM SKAPINSKI

PLEASE SEND INQUIRIES TO:
LIST
MR. HARVEY RAIT
5 PERI LANE
VALLEY STREAM, N.Y. 11581
PLEASE SEND SUBMISSIONS TO:
LISTING
MR. FREDERIC STERN
214 ROBERTS ST.
HOLBROOK, N.Y. 11741

NYTSE

NYTSE MEETS THE MONDAY AFTER THE LIST MEETING AT:
MISS MIMS RESTAURANT
PARK AVENUE SOUTH
BETWEEN 21 ST. AND 22 ST.
MEETINGS START 7:30 PM.

COMING EVENTS:

FEB. 10, 1991 LIST MEETING
FEB. 11, 1991 NYTSE MEETING

MEETING MINUTES
JAN. 6, 1991

HARVEY CALLED THE MEETING TO ORDER AT 2:30PM.

LIST SOLD 2 MORE COPIES OF TECHNICAL TIDBITS, FRED TO SHIP.

JOHN PAZMNO HAS REVISED THE ENROLLMENT FORMS TO REFLECT THE INCREASE IN DUES.

EFFECTIVE JAN. 1, 1991 DUES WERE INCREASED FROM: $15.00 TO: $16.00.

WE RECEIVED MORE INQUIRIES, JOHN TO ANSWER.

SAD NEWS

WE RECEIVED WORD THAT THE OTTOWA T/S USERS GROUP HAS DIS-BANDED.

PROGRAM TAPES

THE LIST TS1000 PROGRAM TAPE IS READY FOR SALE. THE PRICE IS $6.00 BY MAIL, $4.00 WHEN PUR-CHASED AT THE MEETINGS.

A SPECIAL PROGRAM TAPE WILL BE DEVELOPED TO SEND TO NEW MEMBERS. THIS TAPE WIL. BE MADE ON C-10 CASSETTE TAPES. THIS TAPE WILL BE IN 2 VERSIONS, 1 FOR THE TS1000 AND 1 FOR THE TS2068.
IF YOU ARE A NEW LIST MEMBER WHO HAS JOINED WITH IN THE LAST YEAR, SEND A POST CARD TO LIST C/O HARVEY RAIT AT THE ABOVE ADDRESS, AND ADVISE WHICH VERSION YOU WANT.
THIS SPECIAL TAPE WILL BE SENT TO YOU FREE OF CHARGE, COMPLI-MENTS OF LIST.

LIST MERCHANDISE SALE

LIST HAS THE FOLLOWING MERCHAND-ISE FOR SALE, IT IS SELLING FAST SO YOU BETTER HURRY;

TECHNICAL TID-BITS	$4.00
QL MICRODRIVE CARTRIDGE, SET OF 4 IN CASE	$15.00
TS2040-PRINTER	$15.00
TAPE RECORDER-PANASONIC	$15.00
QL MOD. ADAPTER	$20.00

PROGRAM TAPES FOR THE TS1000 OR TS2068. PROGRAMS ARE RECORDED ON QUALITY, NAME BRAND, C-60 CASS-ETTE TAPES. $6.00
THE ABOVE MERCHANDISE CAN BE SEEN AND PURCHASED AT THE NEXT LIST MEETING.

BBS NEWS

ALVARO BRANDON REPORTS THAT THEIR STILL IS A TIMEX ONLY BBS. (SIR CLIVES CASTLE BBS) 613-745-8838 IS RUN BY DAVID SOLLY AND MIKE DOVE. GIVE THEM A CALL AND KEEP THEM ALIVE.

CLASSIFIEDS

THIS CLASSIFIED SECTION IS AVAILABLE TO ALL LIST MEMBERS FREE OF CHARGE.
THE ONLY RESTRICTION IS THAT IT IS TO BE USED ONLY FOR THE SEEKING, SELLING OR SWAPPING OF SINCLAIR, TIMEX OR MICROACE COMPUTER EQUIPMENT, PERIPHERALS AND SOFTWARE.
LISTING, LIST, AND ITS OFFICERS DO NOT ENDORSE, WARRANTY, OR GUARANTEE ANY OF THE ITEMS LISTED IN THIS CLASSIFIED SECTION

I AM LOOKING FOR SOFTWARE TO DRIVE AN AERCO INTERFACE AND GORILLA PRINTER WITH A TS1000. PLEASE CONTACT FRED AT 516-737-0983.

IRISH PCC-10 BLANK TAPES AT AN INCREDIBLY LOW PRICE. CALL TOM 516-732-1825.

A FINAL WORD

MY NAME IS FRED STERN AND I AM THE EDITOR OF THIS EDITION OF LISTING.

THIS ISSUE OF LISTING IS IN MEMORY OF WARREN FRICKE. A FELLOW SINCLAIRIST WHO HAS CON-TRIBUTED MUCH TO SINCLAIR/TIMEX USERS EVERYWHERE.

MANY THANKS TO TOM SKAPINSKI. OUR NEW FRONT COVER AND CHANGED FORMAT ARE THROUGH HIS EFFORTS.

IN THIS NEW YEAR OF 1991, LIST-ING IS PLANNING TO BRING MORE ORIGINAL ARTICLES, CONSTRUCTION PROJECTS, AND PROGRAMS.

ALSO WATCH FOR;
NEW PROGRAM TAPES.
TECHNICAL TID-BITS PART II.
LIST SAVINGS AND LOAD GUIDE.

THE CLASSIC SINCLAIR

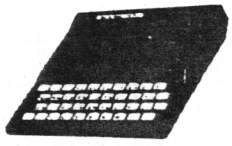
LIST TS1000.1 TAPE

+++++++++++++++++++++++++++++++
THE FOLLOWING IS A LISTING OF
PROGRAMS ON LIST TAPE TS1000.1.

THESE PROGRAMS ARE FOR THE
ZX-81 AND TS1000 WITH THE 16K
RAM MODULE, OR THE TS1500.

THESE PROGRAMS COME ON A QUALIT
C-60 LOW NOISE CASSETTE TAPE.
+++++++++++++++++++++++++++++++---

TO GET THIS TAPE SEND $6.00 TO
LIST C/O Harvey Rait
5 Peri Lane
Valley Stream, NY 11581



MENU

" PSYCHIATRIS "
" MAZEDRAW "
" MORTGAGE "
" ETCHER IS "
" PARADE "
" DRAW "
" WORDSEARCH "
" WEIGHT "
" WORDS "

MULTIPLICATION, GRADE SIX

This tutorial demonstrates the simplicity with which a 64K computer can be programmed to provide a no-frill means of developing an unlimited number of multiplication problems for a math practice session of a six-grader. The program LISTing is short and can readily be converted to the basic language of other computers.

One of the biggest drawbacks of most software packages provided by the schools today for their students is that the programs are too elaborately written. Thru these programs the inanimate computers are expected to teach, but they cannot anticipate a child's needs, and are thus very inefficient as a teaching medium. As a consequence students can spend up to eighty percent of their time standing by.... watching crude computer animations, slow developing and fancy picture embellishments, repetitious displays of the credit and introductory screens, unclear menus, confusingly abbreviated instructions, and the like. Much of this rot is created by the programmers and producers of the software, who appear to be more concerned with demonstrating their programming ability than they are the education of the child. The relatively poor grades that students currently are scoring on standard tests, compared to the grades in the years before computers entered the classrooms, is principally due to this inefficient method of teaching. But, computers can be of help in providing a student with practice exercises to follow up on classroom instruction, by a teacher. H-34 is a program written for this purpose.

Routine H-34 does nothing more fancy than to develop two decimal numbers for multiplication on paper by the student who then enters his answer. The computer then checks it and indicates whether it is correct or not.

There are two subroutines in this program suitable for use in other programs. The one in lines 1000 to 1040 generates decimal and integral numbers in the range of 1 to 900+. They are developed as "strings" so that the individual digits can be manipulated. They are then converted to numbers for computation and display.

The subroutine in 500 to 680 reviews all INPUT and rejects anything that cannot be a valid answer. In other words, it monitors input so that nothing other than positive (or negative) numbers of 2 or more digits get by.

LONG DIVISION, GRADE SIX

Routine H-35 covers problems in long division and is similar in construction to H-34. The two routines could have been combined and supplied with a menu for choosing which operation to play, but I decided against this. Multiplication is usually taught several weeks before long division. They are not taught concurrently. Hence there is little to be gained by a combination.

It is very easy on the TS2068 to convert one similar program into another, as editing is simple on this computer. Starting with the LISTing of H-34, I altered the following lines....

5, 10, 20, 40, 80, 100, & 120

By comparing them you should be able to discern the particular action taken. But line 120 may need a further explanation. In H-34 we compared the student's response "try" with the computer's "ans", and allowed them to differ by no more than .03 in value. In division the difference is not a good criterion as all of the answers will be small. Therefore we used a ratio in the H-35 routine so that to score correctly, the difference between the two values divided by the computer's value agrees to within 6 significant figures. This requires that the student carry his work out to 6 figures, but it also limits his work to this extent.

H-35 has an added subroutine, lines 1500 to 1540, which generate the traditional division symbol which is not on most computers, including the Timex. This symbol, once generated, can be printed out simply by calling for CHR$ 146, or by the letter "A" in graphics. In this tutorial the graphics letter A was used and the symbol printed in its place. One of the many nice features of the Timex is its ability to mix graphics and text at any time and any where on the screen. Most other computers, operating in basic, cannot do this.

You can use this subroutine in other programs and can add up to 19 additional special symbols. More than 20 can be added but an explanation of the method will have to wait for an other day. The first two subroutines can also be incorporated into other programs. They may need to be altered slightly to suit your application.

Warren Fricke

H-34

```
  5 REM ** "I-16", 1-6-90, WF
 10 REM ** MULTIPLICATION,
           GRADE SIX

 20 RANDOMIZE
 30 CLS : LET F=0
 40 PRINT "You are expected to
be within     .03 of the correct
answer to         be marked correc
t."
 50 FOR I=1 TO 10
 60 GO SUB 1000: LET N1=VAL N$
 70 GO SUB 1000: LET N2=VAL N$
 80 LET ans=N1*N2
 90 PRINT '"QUESTION No. ";I
100 PRINT "   ";N1;" X ";N2;" =
";
110 INPUT T$: GO SUB 500
115 LET try=VAL H$: PRINT try
120 IF ABS (ans-try)>.03 THEN G
O TO 150
130 PRINT "   Correct.  Press E
NTER."
140 LET F=F+1: GO TO 160
150 PRINT "   Wrong.  Answer is
";ans: PRINT "    Press ENTER."
160 INPUT z$: NEXT I
170 PRINT '"Your score was ";F;
" out of 10.        Press ENTER t
o play again."
180 INPUT z$: GO TO 30
190
500 REM ** INPUT MONITOR
510 LET H$=T$
515 IF LEN T$<2 THEN GO TO 560
520 IF T$(1)="+" OR T$(1)="-" T
HEN LET T$=T$(2 TO )
530 GO TO 600
560 FOR T=1 TO 120: NEXT T: GO
TO 110
600 FOR T=1 TO LEN T$
610 IF (T$(T)<"0" OR T$(T)>"9")
AND T$(T)<>"." THEN GO TO 110
620 NEXT T
630 LET S=0
640 FOR T=1 TO LEN T$
650 IF T$(T)="." THEN LET S=S+1
660 IF S>1 THEN GO TO 110
670 NEXT T
680 RETURN
690
1000 REM ** NUMBER GENERATOR
1010 LET A$=STR$ (1+900*RND*RND)
+"000"
1020 LET w=2+INT (6*RND)
1030 LET N$=A$(1 TO w)
1040 RETURN
```

```
You are expected to be within
 .03 of the correct answer to
     be marked correct.

QUESTION No. 1
  197.9 X 11 = 2176.9
   Correct.  Press ENTER.

QUESTION No. 2
  3.3 X 354 = 1168.2
   Correct.  Press ENTER.

QUESTION No. 3
  144 X 114 = 16514
   Wrong.  Answer is 16416
   Press ENTER.
```

H-35

```
  5 REM ** "I-17", 1-6-90, WF
 10 REM ** LONG DIVISION,
           GRADE SIX

 20 RANDOMIZE : GO SUB 1500
 30 CLS : LET F=0
 40 PRINT "You are expected to
be correct  in the first 6 signi
ficant fig- ures to be marked co
rrect."
 50 FOR I=1 TO 10
 60 GO SUB 1000: LET N1=VAL N$
 70 GO SUB 1000: LET N2=VAL N$
 80 LET ans=N1/N2
 90 PRINT '"QUESTION No. ";I
100 PRINT "   ";N1;" ÷ ";N2;" =
";
110 INPUT T$: GO SUB 500
115 LET try=VAL H$: PRINT try
120 IF ABS (ans-try)/ans>1E-5 T
HEN GO TO 150
130 PRINT "   Correct.  Press E
NTER."
140 LET F=F+1: GO TO 160
150 PRINT "   Wrong.  Answer is
";ans: PRINT "    Press ENTER."
160 INPUT z$: NEXT I
170 PRINT '"Your score was ";F;
" out of 10.        Press ENTER t
o play again."
180 INPUT z$: GO TO 30
190
500 REM ** INPUT MONITOR
510 LET H$=T$
515 IF LEN T$<2 THEN GO TO 560
520 IF T$(1)="+" OR T$(1)="-" T
HEN LET T$=T$(2 TO )
530 GO TO 600
560 FOR T=1 TO 120: NEXT T: GO
TO 110
600 FOR T=1 TO LEN T$
610 IF (T$(T)<"0" OR T$(T)>"9")
AND T$(T)<>"." THEN GO TO 110
620 NEXT T
630 LET S=0
640 FOR T=1 TO LEN T$
650 IF T$(T)="." THEN LET S=S+1
660 IF S>1 THEN GO TO 110
670 NEXT T
680 RETURN
690
1000 REM ** NUMBER GENERATOR
1010 LET A$=STR$ (1+900*RND*RND)
+"000"
1020 LET w=2+INT (6*RND)
1030 LET N$=A$(1 TO w)
1040 RETURN
1050
1500 REM ** GENERATE ÷ SYMBOL
1510 FOR T=USR "a" TO USR "a"+7
1520 READ w: POKE T,w: NEXT T
1530 DATA 0,24,0,255,0,24,0,0
1540 RETURN
```

```
You are expected to be correct
in the first 6 significant fig-
ures to be marked correct.

QUESTION No. 1
  33 ÷ 77.8318 = 0.423991
   Correct.  Press ENTER.

QUESTION No. 2
  116.6 ÷ 12.9 = 9.03872
   Correct.  Press ENTER.

QUESTION No. 3
  209 ÷ 3.79436 = 55.182
   Wrong.  Answer is 55.081753
   Press ENTER.
```

The Logic Operator, NOT

For many of us, computer logic is a bit difficult to understand, and articles like the one reputed to be by Sharon Z. Aker in the July '88 issue of UPDATE muddies the water more. On page 14, under the title of Priorty, the article states that NOT B<C is interpreted as (NOT B)>C. This is wrong. NOT applies to the entire expression B<C as the < operator has a higher priority than NOT and the computer will evaluate B<C first. Look at page 228 of the manual for a priority listing.

NOT applied to a condition can result only in a logic value of 0 or 1, regardless of the appearance of the condition. For example, consider.....

NOT X = 50

This is interpreted by the computer as....

NOT (X = 50)

If X does equal 50, the condition, X = 50, is TRUE and results in a logic value of 1 for this condition. Then NOT 1 has a logic value of 0.

Conversely, if X has a value of anything other than 50, the condition is considered FALSE, and X = 50 results in a logic value of 0. Then NOT 0, in turn, has a logic value of 1, and the THEN action will take place.

We can show this by a short test program.....

```
10 LET X = 50
20 IF NOT X = 50 THEN PRINT
     "Yes"
30 IF NOT (X = 50) THEN
     PRINT "OK"
40 PRINT "End of test"
```

Now change the value of X in line 10 to anything but 50 and lines 20 and 30 will print out their strings. Why? Because X = 50 is not TRUE and its logic value becomes 0. Now, NOT 0 gives a logic value of 1. As the entire expression has a logic value of 1, the THEN action takes place.

The parentheses in line 30 are not necessary, but if you wish to make the evaluation of an expression clear, use them. The computer simply ignores superfluous parentheses, and evaluates X = 50 first because of the priorty of the = operator.

Let's go a step further. The computer considers all logic on a numerical basis. To the computer, every logic term that is TRUE is assigned the number 1; otherwise it gets a 0. We can put this to a test by adding lines such as the following to our test program.....

```
15 PRINT X = 50
25 PRINT NOT X = 50
35 PRINT NOT (X = 50)
```

and we will get nothing but the numbers 1 and/or 0, on printout of these lines.

You can get the same response by evaluating a logic statement yourself and substituting the resulting logical value in the computer line. Instead of line 20 as written above, write it as.....

```
20 IF 1 THEN PRINT "Yes"
```

and the computer will print the string. Change the 1 to 0 and the computer will not print the string. As we said before, the computer recognizes every thing but 0 as the number 1. So now use something like -3 in place of the 1. Again, the line will print out the string word.

If you get this concept of logic well established in your mind, you will have little or no trouble with logic NOT from here on in.

There is yet another way of treating logic NOT. You can change any expression wherein it appears to an equivalent one by recalling that it merely reverses logic TRUE and FALSE. In doing so, NOT drops out of the expression. So NOT X = 50 can be replaced by X <> 50. Shall we try another? OK. NOT Y > 10 can be replaced by Y <= 10. I personally don't prefer this way of handling NOT because we tend to forget how the computer itself treats this unique operator.

Warren Fricke

8 - 5 - 88

L.I.S.T.

L.I.S.T.

TIMEX LOGIC, PART I

This series of articles is called Timex Logic rather than computer logic because not all computers treat and/or use logic as fully as the Timex computer can. Some of the logic operations described in this series cannot be used on many other computers, including one fruity one, widely pushed onto our public school systems. This is not intended to imply that other computers cannot perform similar tasks. They can do the same thing by devious means but need more programming lines and consume more time in so doing. Appropriate logic can cut a lot of corners.

Line 10, that follows, is a statement containing a single logic condition. The construction of the line is typical of most, but not all, logic statements..... IF "some logic condition" THEN.....

```
10 IF X>5 THEN PRINT "OK"
```

X>5 is read X"greater than" 5, is a logic condition or relation.

X...is a variable that may have any numerical value.

> ..means "greater than" & is one of six symbol operators, =, <, >, <=, >=, and <>. Refer to page 228 of the manual for their meanings.

Line 10 means that if X is greater in value than 5, print the word "OK". Whatever follows THEN is the action to be taken, which can be almost anything. Simple as this statement seems to be, the computer must evaluate the logic condition of X>5 before it can act properly. It does this by making a comparison between X and 5, and then substituting a "logic value of 1" for the condition if the condition is TRUE, or a "logic value of 0" for the condition if it is FALSE. It follows that the condition will be TRUE or FALSE depending upon the value of the variable X at the time the computer works on this line, and the comparison is made.

Although the computer itself assigns a logic value of 1 to a TRUE condition, it considers any number other than 0 to be logical TRUE, even negative numbers. Thus a logic value of 0 denotes a FALSE condition and any other logical value denotes a TRUE condition. Hence, logic is number oriented. We can test this by adding a program line..

```
5  LET X=6
```

If we now RUN this two-line program, line 10 will print out the word "OK". This is because the logic condition of X>5 is TRUE, and the computer substituted a logic value of 1 for the condition. By direct entry have the computer PRINT X>5 and the response will be 1, the logic value of a TRUE condition.

If we change the value of X in line 5 to say 3 and RUN the program again, nothing prints out. This is because the computer assigned a logic value of 0 to the X>5 condition, as it is now FALSE. As before, and by direct entry, ask the computer to PRINT X>5. The response will be 0, the current logical value of X>5.

Now DELETE line 5. And in line 10 in place of the condition X>5, substitute 1, the logic value for a TRUE condition. RUN the one-line program and the word "OK" will print. It will also print "OK" no matter what numerical value is substituted for X>5, even negative values. Try some. But if a 0 is substituted for X>5, indicating a FALSE condition, nothing prints out.

It is important to realize that the conditions referred to as TRUE or FALSE do not involve morals nor are they indicative of desireability in any way. They are merely terms handed down from a branch of mathematics that was in existance long before the Timex computer. You can verify this by running thru the preceding exercises with the condition changed to read X<5, or X=5, etc.

NOT is unique as a logic operator. It is the only one of them that is also a function; so it always has a following argument (condition) and a subsequent result, which in this case is always 0 or 1. Unlike other functions its priorty is a low 4, ranking it below the six symbol operators and just above AND and OR.

TIMEX LOGIC, PART II

In the preceding chapter we learned that logic is number oriented, and that conditions which are TRUE are replaced by the computer with a logic value of 1, and conditions which are FALSE are replaced by a logic value of 0. Program lines with statements containing a single logic condition usually present no problem to understand, either to devise when creating a program, or to determine their effect in one that is already existing.

But there are numerous occasions where two or more conditions must be acted upon jointly before the computer can take proper action. This is done by appropriately connecting the logic conditions with the logic operators, AND and OR.

AND can be likened to connecting two logic conditions in series, so that both conditions must be considered, one and then the other, before proper action can be taken. Such a statement might appear as.....

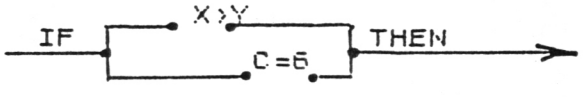
```
IF X>Y AND C=5 THEN....
```

In a diagram arrangement this same statement can be shown as....

In such a diagram we might consider computer functioning to be similar to the flow of current in an electrical circuit. The two conditions might then be analogous to switches in the circuit, closed if a condition is TRUE and open if it is FALSE. So visually, this diagram shows that AND requires both conditions to be TRUE for current to flow and the THEN action to take place.

OR, in a statement, can be likened to connecting two logic conditions in parallel so that if either condition is TRUE, its branch conducts and the THEN action takes place. Such a statement and its corresponding diagram might look like the following example.....

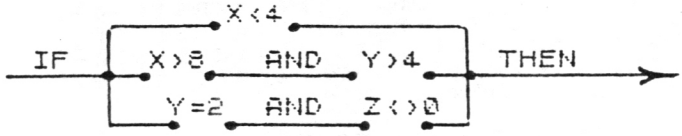
```
IF X>Y OR C=5 THEN....
```



Statements may become more complex than these, but no matter how many logic conditions there may be, such a statement may always be diagramed by groupings of series and parallel circuits. The resulting diagram will be relatively easy to analyze. Where both AND and OR conditions occur in a statement, connect the AND conditions first as AND has a higher priorty than OR. Consider the following statement.....

```
IF X<4 OR X>8 AND Y>4 OR Y=2
   AND Z<>0 THEN.....
```

This line would be diagramed by first connecting the AND conditions into series branches and then connecting the various branches in parallel. It would end up like.....

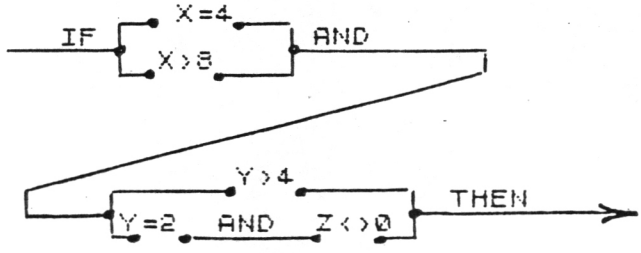


From this diagram one can readily see that the THEN action will take place if any one of the three branches contains nothing but TRUE conditions, regardless of the status of the other two branches. For example, if X is equal to 3 (refer to the top branch), the THEN action occurs.

Parentheses, having a high priorty of 16, will affect the logic of a statement in which they appear. Priorty requires that the computer consider them first and the resulting diagram should show this. Insert some parentheses at random into the previous statement and it might look like.....

```
IF (X<4 OR X>8) AND
   (Y>4 OR Y=2 AND Z<>0) THEN..
```

Connecting the elements within the parentheses first, we get this diagram.....



Now it should be apparent at a glance that there are 4 paths by which current can flow; thus four possible combinations of conditions that will produce a THEN action.

TIMEX LOGIC, PART II

In the use of diagrams, one can work backwards from a diagram to produce a statement of logic that will cover any conceivable set of conditions. It can be of considerable help to a programmer. It eliminates uncertainty and guess work, and perhaps the need for a lot of trial and error testing.

Although we used numerical conditions throughout these illustrations (because they need less space to show), conditions containing STRING$ are very common and can also be TRUE or FALSE. They would form diagrams in a similar manner. Later in this series we will encounter some.

This might be as good a place as any to inject a word of caution when using a condition such as Y=2 in the preceding statement. If the value of the variable Y is to be computed by the program, it may show up with a value of say 1.99999999, depending upon the factors used and the arithmetical operations which produce it. The computer does not consider such a value to be equivalent to 2, in a comparison. So you might want to write the condition as..... INT(Y+.1)=2, or what ever else might be needed to avoid a faulty comparison.

In the next chapter, we will explore other Boolean logic arrangements, and some others besides these that are uniquely Timex.

Warren Fricke
8-12-88

TIMEX LOGIC, PART I

NOT is used principally to inverse the logic value of another logic operation. In expressions like NOT X=Y, if X=Y is TRUE (logic value 1), NOT X=Y is FALSE (logic value 0). If X=Y is FALSE, then NOT X=Y is TRUE. In this combination, the Timex computer evaluates the operation X=Y first, as the logic symbol = has a priorty higher than NOT. Parenthes (high priorty) around the operation X=Y are not needed to do this, but may be used sometimes for clarity. The Timex computer simply ignores superfluous parentheses.

NOT can be eliminated, if desired, by inversing the expression that it modifies. For example, NOT X=Y can be replaced by X<>Y. But use care in so doing. The inverse of X>Y is X<=Y, and not simply X<Y.

There are times when NOT can not conveniently be eliminated. For example, NOT X means that if X has a value of 0, NOT X has a logic value of 1. If X has a value other than 0, NOT X has a logic value of 0. From this one can appreciate that NOT "something" will always have a logic value of 0 or 1, and nothing else.

In Part II we will show how compound logic statements can be built up using the combining logic operators, AND and OR. And, how the Timex computer considers the priorty of all operations, including those assigned to logic conditions, so that it can do "first things first"; thus arriving at the proper action to take in any situation.

Warren Fricke
8-9-88

ZX-81 AND TS1000

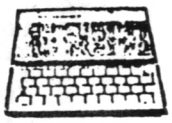
TECHNICAL TIDBITS

SECOND PRINTING

On Sale

$4.00

TIMEX LOGIC, PART III



It has been said here before that the computer is a number oriented machine. Every thing that it does, it does by the manipulation of numbers. Basic is a language akin to English, devised solely for the benefit of us humans. The computer itself doesn't need Basic, and early models of computers did not have it. All of the Basic commands, operations, symbols, etc, used by programmers are converted immediately into code numbers by the computer for its own use. Then to communicate back to us, the computer kindly replaces the numbers with a language that we can more readily understand. In an 8-bit computer such as the Timex, there are only 256 distinct numbers in a byte, from 0 to 255, but these numbers have different meanings to the computer, depending upon how and where they are used. That's why, for example, the number 201 can represent RETURN in a machine code instruction, and the symbol <> used in a logic condition of Basic. That's why 0 in logic is set aside to mean FALSE, and any other number in similar circumstances signifies TRUE.

Another computer trait we wish to emphasize again is the way it responds to priority. Priorities are listed on page 228 of the manual. Add parentheses, priority 16, to this list. Every thing that the computer does, it does in the order dictated by priority. Basic statements, including logic, are scanned left to right on a program line, instructions followed according to priority, highest first, then the next highest, until all operations are considered.

Let's put these two logic considerations to a test with the following short program.....

```
 5 LET Y=2
10 PRINT 5 + Y*6
```

Look at line 10. It doesn't contain a logic condition, but priority is needed to perform the mathematics at the proper time. The * symbol has priorty higher than the + symbol, so the computer does the multiplication first, and then adds the result to 5. The answer, 17, will appear if you RUN the program.

Next substitute an = mark for the + sign in line 10.....

```
 5 LET Y=2
10 PRINT 5 + Y=6
```

Now there is a logic condition in line 10, but the + sign has a higher priorty than the equal symbol, so the computer adds 5 to 2 and gets 7. Then it looks at the condition 7=6. It compares the two numbers and finds that they are not equal. So it prints out a 0, the logic value of a FALSE condition. Change the 6 in line 10 to a 7 and RUN the program again. You will now get a printout of 1, indicating that the logic value of 7=7 is 1, indicating a TRUE condition.

Consider the manner in which the Timex computer evaluates or alters the value of a variable. Seemingly this construction may look like a departure from the priority rule. Enter the following lines.....

```
 5 LET T = 5
10 LET T = T+3
15 PRINT T
```

Meeting this for the first time, one might conclude that this is arithmetically incorrect. How can a number be equal to itself plus 3? Here literal meaning gives way to computer talk, and it means change T to be equal to itself plus 3. In this situation the = mark is an "assignment operator", having a priority of 1, placing it at the very bottom of the priority scale. In another sense LET T= can be considered a syntax arrangement, a grouping of characters that perform a particular function last and independant of the other operations in the statement. Now change line 10 of this little program to read.....

```
10 LET T = T = T
```

RUN it and the answer comes out 1. This is because the computer considers this line to be equivalent to...

```
10 LET T = (T=T)
```

Now (T=T) is true for all values of T; so it has a logic value of 1. Hence T itself becomes 1. Go a step further and add another = T....

```
10 LET T = T = T = T
```

Run this and the answer comes up 0. This is because the computer evaluates line 10, in stages, as follows.....

```
10 LET T=(T+T)=T, which is...
10 LET T=( 1 )=T, which is...
10 LET T= (1=T) , which boils
     down to....
10 LET T= 0
```

And the final answer will be 0 no matter how many times we add an =T. LET appears in many lines containing logic conditions. Try the following in our little program....

```
10 LET T = T = 50
```

Since T has a value of 5 before getting to this line, the condition T = 50 becomes a logical 0 and T finalizes to a value of 0 also. If T had a value of 50 when the program got to this line, the condition would be TRUE, logic value 1, and T ends up with a value of 1 also. Thus the final value of T will be either 0 or 1 depending upon its prior value.

Appropriate logic expressions enable the computer to make quick decisions with a paucity of words or characters. To the programmer this means less bytes are required, conserving RAM. Short, concise statements also RUN faster as a rule, another asset. Experienced programmers build up a repertoire of logic short cuts. Consider a situation where one might want to develop either a +1 or a -1, at random. Useful if back and forth, up and down motions are required. An expression like the following will do it....

```
LET K = 1 - 2*( RND > .5)
```

The condition in parentheses will be TRUE half of the time and FALSE the other half. So it might have a logic value of 1 or 0. From here on the procedure is pure math. A logic value of 1 makes K = -1, and a logic value of 0 makes K= +1. By the way, did you notice that the parentheses are necessary? Otherwise the computer will consider the * operator before the > symbol. There are short cuts galore, especially in Timex logic; as we shall see.

Consider an expression like IF B<>0.... It can be replaced simply by IF B. This is because any value that B would have other than 0 would be considered logical TRUE. The same goes for an expression like...

```
IF 17 * n <> 0, which is the
     same as...
IF 17 * n
IF TRUE.... (n<>0)
```

We can have other mixtures of arithmetic and logic operations and priority will decide the outcome. Consider this one....

```
 5 LET D = 6: LET E = 5
10 PRINT D < 8 - E
```

Before you RUN this, take a try at the answer, remembering that the negative sign has a higher priority than the < mark causing the computer to consider the condition to be 8 - 5 or 3. As 3 is not larger than 6, the entire expression is FALSE. Hence a logic 0 will PRINT.

In the fourth part of this series we will look at some logic expressions not available on other computers. Expressions that enable Timex programming to be more concise when the occasion arises, and they can be used.

Warren Fricke
Revised 10-28-89

ZX-81 AND TS1000

TECHNICAL TIDBITS

SECOND PRINTING

$4.00

LISTinG Policy

Annual Dues.....$16.00

One "Sample" copy sent upon receipt of large SASE.

Copies Provided on exchange basis with other bona fide user groups.

L.I.S.T.InG is published monthly by LIST (Long Island Sinlair Timex) Group, a non-for-profit users group.

NOTE:
Normal membership year is Feb. through Jan. at a cost of U$15.00. By keeping as many members as possible on that basis, we keep our costs and chances of error down. If you wish to begin your subscription later in the year, please sign up for the end of this year and all of next.

We will accept partial years or different sucription runs, on a limited basis (particularly from members outside the U.S.). But please be aware that in addition to possible rate increases, your "account" must be handled "by hand" and errors may occur.

POLICY REGARDING CONTRIBUTED MATERIAL:
We are always looking for interesting articles, programs, reviews etc. to keep our members informed and entertained. Articles submitted for publication are printed on the following basis :

1) You , the writer, maintain the full copyright and can resell, lend or give away your work, as you wish.
2) We are granted the right to publish your material, in the original issue in which it appears. Reprints (e.g., to supply orders for back issues) will include your material as a part of its original issue.

We can't (for now) pay you for your material, but you will receive a copy of the issue in which it is published, even if you are not a member. You may get more than one issue and you will definitely earn the respect and appreciation by your grateful peers.

Articles represent the opinion of the author and not necessarily the LIST Group. LIST disclaims any responsibility for any thing you may do to your computer as a result of reading any articles in LISTinG.

L.I.S.T.
5 PERI LANE
VALLEY STREAM, NY
11581

LONG ISLAND NY 117
PM
14 JAN
1991

Antonello, National Gallery

TO Don Lambert JAN/92
1301 Kibinger PL
Auburn, IN
46706-3010

FIRST CLASS MAIL
DATED MEETING NOTICE

UPPER RIGHT
CORNER OF
YOUR LABEL
IS DATE OF
LAST ISSUE.